REMARKS

The Examiner has rejected Claims 1-90 under 35 U.S.C. 112, second paragraph, as being indefinite. Such rejection is deemed moot in view of the clarifications made hereinabove to the claims.

The Examiner has rejected Claims 1-94 under 35 U.S.C. 103(a) as being unpatentable over Uszok et al. (U.S. Publication No. 2004/0205772) in view of Kouznetsov et al. (U.S. Patent No. 6,931,546). Applicant respectfully disagrees with such rejection, especially in view of the amendments made hereinabove to the independent claims. Specifically, applicant has amended the independent claims to at least substantially include the subject matter of former dependent Claims 8, 9, 10, 93, 94 et al.

With respect to the independent claims, the Examiner has relied on paragraphs 0014, 0050, and 0055 from the Uszok reference to make a prior art showing of applicant's claimed technique "receiving code to receive at said destination computer operation specifying XML data sent by said source computer" (see this or similar, but not necessarily identical language in the independent claims).

Applicant respectfully asserts that the excerpts from Uszok relied upon by the Examiner discloses a technique where "[t]he client side--mBot--implements a presentation layer, active component or even pure XML or HTML, and may have multiple presentations for different platforms" (emphasis added). However, having the client side implement pure XML simply fails to meet a "receiving code to receive at said destination computer operation specifying XML data sent by said source computer" (emphasis added), as claimed by applicant.

In the latest Office Action dated 03/08/2006, the Examiner has further argued that "the plug-ins can be passed from the botServer manager to plug-in manager, which is the target process" and that "[b]oth managers are performed at the same target computer

botServer." Additionally, the Examiner cited the following excerpts from Uszok to support such assertion.

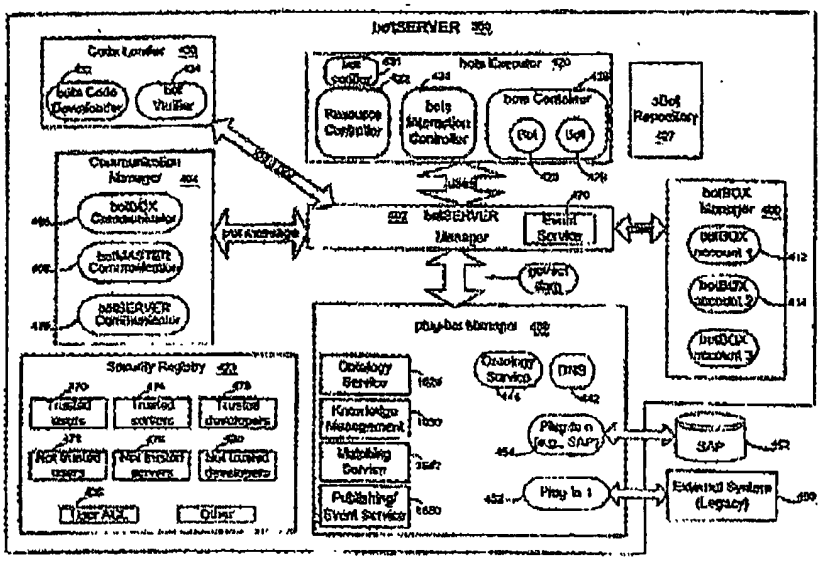

Figure 4

(Uszok, Figure 4)

"[0079] Plug-ins: The botServer standard services can be extended by a mechanism of plug-ins. **A plug-in is a software component installed by botServer administrator within the plug-ins manager (450) which communicates with other plug-ins and bots via the botServer manager 402 using messages.** From the usability point of view, a plug-in is somewhat similar to a bot--it has an unique ID, understands bot messages, participates in protocols etc. A plug-in, however, never migrates, cannot be automatically downloaded and installed, and usually has much higher system security permissions. A plug-in also may possess any number of system threads (since it doesn't have to strictly adhere to a request-response execution schema). Plug-ins are usually used to interface with various external systems, perform demanding real-time operations and provide other services. For example, FIG. 4 illustrates a plug-in 454 for interfacing with an SAP server 462. Another plug-in 452 interfaces with a legacy system 460 such as an enterprise database running on a mainframe. FIG. 17 shows auction and B2B plug-ins on botServers.

[0080] Plug-ins may be configured so that they have the same ID on every server. That strategy relieves bots from having to search for an appropriate plug-in if they know the requested plug-in ID. Some standard system services (e.g. Ontology service 1620 or botDNS 442) may be implemented as plug-ins as well. Specialized plug-ins can be created by botServer owners to provide services that make the server more attractive to bots (e.g. in order to advertise some other profitable services). Other examples shown in FIG. 4 (and FIG. 16) are knowledge

> management service 1630, matching service 1642 and a publishing/event service 1650. Generally the plug-ins should be created to be compatible with botServer scaling." (Uszok, Paragraphs 0079, 0080 - emphasis added)

Applicant respectfully asserts that the additional excerpts from Uszok relied upon by the Examiner simply disclose that "[a] plug-in is a software component installed by botServer administrator within the plug-ins manager (450) which communicates with other plug-ins and bots via the botServer manager 402 using messages" (emphasis added). However, the "get/set data" and "uses" flows that communicate with the plug-ins and bots "using messages" in Uszok's Figure 4 fails to disclose "receiving code to receive at said destination computer operation specifying XML data sent by said source computer" (emphasis added), as claimed by applicant. In addition, Uszok disclosure that "[b]oth managers are performed at the same target computer botServer" (emphasis added) simply fails to meet "receiving code to receive at said destination computer operation specifying XML data sent by said source computer" (emphasis added), as claimed by applicant.

In addition, with respect to the independent claims, the Examiner has relied on the following excerpt from the Uszok reference to make a prior art showing of applicant's claimed "parsing code to parse said operation specifying XML data to identify one or more complex data types within said operation specifying XML data" and "matching code to match each complex data type with an associated execution process available to said destination computer" (see this or similar, but not necessarily identical language in the independent claims).

> "[0057] Returning now to the bot code acquisition process, the botServer 350 may already have a copy of the corresponding sBot class for operation with the mBot. (Indeed, the botServer 350 could even be hosted on the same site as the bot developer site 310.) If so, the botServer 350 creates an instance of the requested sBot and assigns an identifier to it corresponding to the mBot that made the request. If the appropriate sBot code is not already present, the botServer 350 obtains the appropriate sBot code from the bot developer site 310 by download link 362. Authentication of that code, security issues, payment and other particulars are discussed later in this description. At the botServer, after the sBot is installed, a message originating from botMaster 320 is communicated to the sBot code to initialize

> it to carry out the task requested by the user." (Uszok, Paragraph 0057 - emphasis added).

Applicant respectfully asserts that such excerpt from Uszok relied upon by the Examiner merely teaches that the "botServer 350 creates an instance of the requested sBot and assigns an identifier to it corresponding to the mBot that made the request" (emphasis added). Additionally, the excerpt from Uszok discloses that "[a]t the botServer, after the sBot is installed, a message originating from botMaster 320 is communicated to the sBot code to initialize it to carry out the task requested by the user" (emphasis added). However, creating an sBot instance and initializing it from "a message originating from botMaster" simply fails to meet a technique of "parsing code to parse said operation specifying XML data to identify one or more complex data types within said operation specifying XML data" (emphasis added) and "matching code to match each complex data type with an associated execution process available to said destination computer" (emphasis added), as claimed by applicant.

In the latest Office Action dated 03/08/2006, the Examiner has further argued that "Uszok states matching mechanism (0133) and the XML-based interaction protocols define a set of states that a bot may exist in, rules of transition from one state to another and a description of the schema of data that can be exchanged between participating parties in order to transition from one state to another (128)." Additionally, the Examiner cited the following excerpts from Uszok to support such assertion.

> "[0128] A meeting place in the present system is a separated logical area of the botServer, where bots can interact with chosen types of other bots and plug-ins according to defined interaction protocols. Interaction protocols define a set of states that a bot may exist in, rules of transition from one state to another and [optionally] a description of the structure (schema) of documents (data) that can be exchanged between participating parties in order to transition from one state to another. An interaction protocol can be described, for example, in an XML-based language. The selected protocols are implemented on bot's and plug-ins in accordance with the present invention." (Uszok, Paragraph 0128 - emphasis added).
>
> '[0133] Matching is a general mechanism that enables selection of abstract "offers" that most closely match abstract "requests." The mechanism is abstract, because it does not define what the offer really is. The determination is based only on its

> description and applying sophisticated Al algorithms to actually perform matching, however XML-based, "hard"--regular expression or SQL-like based, matching might be used as well (whichever more useful in a given situation). Such an approach allows, e.g., selecting botServers providing most appropriate services, or selecting trade offers closest to the buyer's requirements--all using the same mechanism. Generally the matching service should be able to provide a match of request and offer even if they cannot be matched by simple comparison.' (Uszok, Paragraph 0133 - emphasis added).

Applicant respectfully asserts that the matching disclosed in the excerpts from Uszok relied upon by the Examiner teach that a "determination is based only on its description and applying sophisticated Al algorithms to actually perform matching, however XML-based, "hard"--regular expression or SQL-like based, matching might be used as well" (emphasis added). Generally disclosing XML-based matching fails to meet "matching code to match each complex data type with an associated execution process available to said destination computer" (emphasis added), as claimed by applicant. Further, Uszok's disclosure where "an interaction protocol can be described, for example, in an XML-based language" simply fails to meet "parsing code to parse said operation specifying XML data to identify one or more complex data types within said operation specifying XML data" (emphasis added), as claimed by applicant.

Further, with respect to the independent claims, the Examiner has relied on the following excerpt from the Kouznetsov reference to make a prior art showing of applicant's claimed technique "wherein said execution process maps configuration data specified within said operation specifying XML data to a configuration data store of said destination computer; wherein said configuration data store is one of: a Windows Registry entry; an INI file; a DAPI store; and a database entry" (see this or similar, but not necessarily identical language in the independent claims).

> 'The above limitations of the prior art are addressed by a system, method and software in which a process is run on a client machine having sufficient privileges to execute privileged processes. This process has a role of a "local system" and is effectively an administrator for the machine. An agent program running in user-mode provides a generic interface. The agent includes an application programming interface ("API") for receiving requests for privileged processes. The agent includes

> an interface to the privileged process as well. The agent includes methods for authenticating any received requests and will only forward a request to the privileged process upon determining that the requesting application has sufficient trust. Hence, the agent provides a level of indirection in accessing the privileged process so that the local system interface is not exposed directly to untrusted entities.
>
> Briefly stated, the present invention involves a system for providing application services in a computing environment having both user-mode processes and privileged-mode processes. An agent executes in privileged mode and exposes an interface to user-mode processes. A user-mode component is provided with an interface configured to access the agent's exposed interface. **A configuration component specifies a list of installable code components that are authorized for installation, wherein the agent will only execute privileged-mode functions in response to accesses by the user-mode code component when the installable code component is represented on the list.'** (Kouznetsov, Col. 4, lines 25-54 - emphasis added)

Applicant respectfully asserts that the above excerpt from Kouznetsov relied upon by the Examiner teaches that an 'agent includes an application programming interface ("API") for receiving requests for privileged processes' (emphasis added). In addition, Kouznetsov discloses a "configuration component [which] specifies a list of installable code components that are authorized for installation, wherein the agent will only execute privileged-mode functions in response to accesses by the user-mode code component when the installable code component is represented on the list" (emphasis added). However, disclosing an API with a configuration component simply fails to meet applicant's claimed technique "wherein said execution process maps configuration data specified within said operation specifying XML data to a configuration data store of said destination computer" (emphasis added), in the context claimed. In addition, Kouznetsov's disclosure of "[a] configuration component specif[ying] a list of installable code components" (emphasis added) in no way meets a technique "wherein said configuration data store is one of: a Windows Registry entry; an INI file; a DAPI store; and a database entry" (emphasis added), as claimed by applicant.

Additionally, with respect to the independent claims, the Examiner has relied on the following excerpt from the Uszok reference to make a prior art showing of applicant's claimed technique "wherein an identifier of an execution process within said complex

data type includes at least one of: data specifying a computer file to trigger said execution process; data specifying a communication channel to trigger said execution process; and data specifying an operating system command to trigger said execution process" (see this or similar, but not necessarily identical language in the independent claims).

> "[0068] Referring again to FIG. 5, establishing a botBox is initiated by the user or prompted by botMaster through the GUI 520. The GUI communicates via the botMaster core 530 to a botBox proxy component 542. **The botBox proxy component initializes local botBox storage 536 through the botMaster configuration component 534 and then utilizes a communication manager 544, and more specifically a botBox communicator component 546, to send a message to request initialization of a corresponding botBox on a botServer.** Any botServer (with botBox facilities) can be used. Preferably, the user will want to select a botServer with high availability if the user bot(s) need to operate around the clock. Bots with more modest requirements can have their botBoxes hosted at more modest botServers, or even on their own home PC. The user can relocate botBox later if desired, and could store it (including all bot information) on machine-readable media for subsequent uploading on another platform. For commercial applications, botBoxes should be housed on a reliable server, preferably independent of the user platform. This architecture supports operation of the user's bots while the user is off-line, and ensures the user's privacy." (Uszok, Paragraph 0068 - emphasis added)

Applicant respectfully asserts that the above excerpt from Uszok relied upon by the Examiner discloses that "[t]he botBox proxy component initializes local botBox storage 536 through the botMaster configuration component 534 and then utilizes a communication manager 544 ... to send a message to request initialization of a corresponding botBox on a botServer" (emphasis added). However, "send[ing] a message to request initialization of a corresponding botBox" fails to meet a technique "wherein an identifier of an execution process within said complex data type includes at least one of: data specifying a computer file to trigger said execution process; data specifying a communication channel to trigger said execution process; and data specifying an operating system command to trigger said execution process" (emphasis added), as claimed by applicant.

In addition, with respect to independent Claim 16, the Examiner has relied on paragraphs 0014, 0050, and 0055 from the Uszok reference to make a prior art showing

of applicant's claimed "data forming code to form at said source computer operation specifying XML data containing one or more complex data types" (see this or similar, but not necessarily identical language in the independent claims).

Applicant respectfully asserts that the excerpts from Uszok relied upon by the Examiner discloses a technique where "[t]he client side--mBot--implements a presentation layer, active component or even pure XML or HTML, and may have multiple presentations for different platforms" (emphasis added). However, having the client side implement pure XML simply fails to disclose the technique of "data forming code to form at said source computer operation specifying XML data containing one or more complex data types" (emphasis added), as claimed by applicant.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir.1991).

Applicant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art references, when combined, fail to teach or suggest all of the claim limitations, as noted above. Nevertheless, despite such paramount deficiencies and in the spirit of expediting the prosecution of the present application, applicant has amended the independent claims to at least substantially include the subject matter of former dependent Claims 8, 9, 10, 93, 94 et al.

With respect to the subject matter of former Claim 9 et al. (now at least substantially incorporated into the independent claims), the Examiner has relied on the

following excerpt from the above Uszok reference to make a prior art showing of applicant's claimed technique "wherein said result data includes data specifying existing configuration data of said destination computer" (see this or similar, but not necessarily identical language in the independent claims).

> "[0122] **The user may choose to create the dynamically created, temporary profiles based on existing ones.** The temporary profile is used for a limited amount of time (specified by timeout, response type, or any other kind of trigger) or per-transaction and after that it is destroyed. This allows the user to block all the later communication attempts (especially spam) with a particular profile--they will fail because the profile does not exist anymore. It also supports the anonymity in the bot system--it is difficult (if possible at all) to track a person using the dynamic profile. The user may also choose to define the information filters on each user profile. Any information flow going through the profile may be analyzed (e.g., checking the sender, analyzing the message content, etc.) and rejected if does not fit to the rules defined in the filter. This allows an additional anti-spam mechanism to be built into the user profiles. The rejected information is not routed to (and from) the user account (optionally only notifying it about the information rejection)" (Uszok, Paragraph 0122 - emphasis added)

Applicant respectfully asserts that the excerpt from Uszok above relied upon by the Examiner merely teaches that "[t]he user may choose to create the dynamically created, temporary profiles based on existing ones." However, there simply is no mention in the above excerpt of a technique "wherein said result data includes data specifying existing configuration data of said destination computer" (emphasis added), as claimed by applicant.

In addition, with respect to the subject matter of former Claim 10 et al. (now at least substantially incorporated into the independent claims), the Examiner has relied on the following excerpt from the above Uszok reference to make a prior art showing of applicant's claimed technique "wherein said execution process maps existing configuration data of said destination computer stored within said configuration data store of said destination computer to said result data to be returned to said source computer" (see this or similar, but not necessarily identical language in the independent claims).

"[0100] FIG. 11-B illustrates a master-slave arrangement for sharing a botBox. Here, the user installs a botMaster application 1118 on a portable device such as a PDA, cell phone, or automobile Internet appliance. This may be a "thin client" with a simplified user interface (perhaps with limited graphics), and it **may lack the software and memory necessary for creating and maintaining a user model.** This botMaster 1118 need not establish its own botBox account. Rather, the "thin" botMaster can interact with the same botBox 1112 that was established by the office botMaster 1110. **The portable botMaster, in a slave mode, employs the user model maintained by the master botMaster 1110 and the portable program assigns to its mBots one of the user profiles made available on the shared botBox 1112.** Creating new profiles or modifying existing ones can be done using the more full-featured office botMaster 1110." (Uszok, Paragraph 0100 - emphasis added)

Applicant respectfully asserts that the excerpt from Uszok above relied upon by the Examiner teaches that "[t]he portable botMaster, in a slave mode, employs the user model maintained by the master botMaster 1110 and the portable program assigns to its mBots one of the user profiles made available on the shared botBox 1112" (emphasis added) since "it may lack the software and memory necessary for creating and maintaining a user model" (emphasis added). The disclosure of "employ[ing] the user model maintained by the master botMaster" simply fails to even suggest a technique "wherein said execution process maps existing configuration data of said destination computer stored within said configuration data store of said destination computer to said result data to be returned to said source computer" (emphasis added), as claimed by applicant.

With respect to the subject matter of former Claims 93 and 94 (now at least substantially incorporated into the independent claims), applicant respectfully notes that, in the latest Office Action dated 03/08/2006, the Examiner failed to specifically respond to these claims presented in Amendment A mailed 12/12/2005. Applicant has reviewed the Uszok and Kouznetsov references in their entirety and asserts that such claim language is simply not met by such references, either alone or in combination.

Thus, a notice of allowance or specific prior art showing of each of the foregoing claim elements, in combination with the remaining claimed features, is respectfully requested.

Thus, all of the independent claims are deemed allowable. Moreover, the remaining dependent claims are further deemed allowable, in view of their dependence on such independent claims.

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 505-5100. The Commissioner is authorized to charge any additional fees or credit any overpayment to Deposit Account No. 50-1351 (Order No. NAI1P480).

Respectfully submitted,
Zilka-Kotab, PC.

Kevin J. Zilka
Registration No. 41,429

P.O. Box 721120
San Jose, CA 95172-1120
408-505-5100